



Autoencoders Explained

5 min read · Jun 16, 2024



Omkar Hankare

Follow



Listen



Share

Part 3: KL-divergence

Sparse Autoencoders: They are a variant of traditional autoencoders, which learn to compress data into a lower-dimensional representation by adding a sparsity constraint to the loss function.

One common approach to achieving sparsity is to add an L1 regularization term to the loss function, which encourages the absolute values of the encoding layer activations to be small. This has the effect of pushing many of the activations to be exactly zero, resulting in a sparse representation.

Another approach to achieving sparsity is to use a special type of activation function, such as the rectified linear unit (ReLU), which is non-linear but also has a sparse output. ReLU sets any negative values in the output to zero, effectively encouraging sparsity in the activations. Yet another approach is to use a combination of both L1 regularization and a sparse activation function.

KL-divergence (Kullback-Leibler divergence) regularization — Kullback-Leibler (KL) divergence is a measure of the difference between two probability distributions i.e It quantifies how much one probability distribution differs from another probability distribution. KL divergence is often used in machine learning as a loss function to train models, especially in generative models where one distribution is the true distribution and another is the estimated distribution generated by the model. The KL divergence loss term in VAEs encourages the encoder network to generate a latent distribution that is close to a known distribution, such as a unit

Gaussian distribution. The closer the encoder's distribution is to the known distribution, the lower the KL divergence loss will be. This encourages the encoder to generate a more meaningful latent space that can be used by the decoder to generate new data points. The KL divergence between two distributions Q and P is often stated using the following notation: $KL(P \parallel Q)$

It is pronounced as “the KL divergence of P from Q” or “the Kullback-Leibler divergence from Q to P”. In this notation, P is usually the true distribution, and Q is the estimated distribution. In the context of multiclass classification, KL divergence is commonly used as a loss function in neural networks to measure the dissimilarity between the predicted probability distribution and the true distribution of the labels. This can help the model to learn to predict more accurate probability distributions for each class. The formula for KL divergence in this context is:

$$KL(P \parallel Q) = \sum_{i=1 \text{ to } N} P(i) \log [P(i) / Q(i)]$$

Here, P is the true probability distribution of the labels (i.e., the one-hot encoded representation of the correct label), Q is the predicted probability distribution of the labels, and N is the number of classes.

For example, let's say we have a multiclass classification problem with 5 classes, and our true distribution (P) for a particular data point is [0, 0, 0.2, 0.8, 0], meaning the data point belongs to class 3 with probability 0.2 and class 4 with probability 0.8.

Now, let's say our model predicts a distribution (Q) for the same data point as [0.1, 0.1, 0.3, 0.4, 0.1]. We can calculate the KL divergence between the true and predicted distributions as follows:

$$KL(P \parallel Q)$$

$$= (0 * \log(0/0.1) + 0 * \log(0/0.1) + 0.2 * \log(0.2/0.3) + 0.8 * \log(0.8/0.4) + 0 * \log(0/0.1))$$

$$= 0 + 0 + 0.2 * \log(0.2/0.3) + 0.8 * \log(0.8/0.4) + 0 = -0.0364 + 0.1178 + 0$$

$$= 0.0814$$

So the KL divergence between the true and predicted distributions is 0.0814. We can interpret this as the amount of information lost when approximating the true distribution with the predicted distribution. The lower the KL divergence, the better the predicted distribution matches the true distribution.

Get Omkar Hankare's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

Subscribe

On the other hand, in the context of generative models such as variational autoencoders (VAEs), KL divergence is used as a regularization term to encourage the learned distribution of the latent space to be similar to a known distribution. This helps to ensure that the generated samples from the model are more diverse and meaningful.

The formula for KL divergence in this context is:

$$KL(P || Q) = -0.5 * \sum(1 + \log(\sigma^2) - \mu^2 - \sigma^2)$$

$KL(P || Q) =$

In this formula, the KL divergence loss term is calculated as the difference between two distributions: the approximate distribution $q(z|x)$ generated by the encoder and the true distribution $p(z)$. The formula measures how much information is lost when using the approximate distribution to represent the true distribution. In the context of VAEs, the approximate distribution refers to the distribution of the latent variables (also called the latent code or latent space) generated by the encoder network. This distribution is denoted by $q(z|x)$, where x is the input data and z is the latent variable. The true distribution $p(z)$ is usually assumed to be a unit Gaussian distribution, which means that it has a mean of 0 and a variance of 1. Here, μ and σ are the mean and standard deviation of the encoded latent variable z , respectively. The sum is taken over all elements of z . here's an example of how the sum is taken over all elements of z in the context of VAEs:

If the latent space is 10-dimensional, then the mean vector and standard deviation vector will each have 10 elements. Let's say the mean vector (μ) is [0.2, -0.5, 1.3, -0.9, 0.7, 0.1, -1.1, 0.4, 0.8, -0.3] and the standard deviation vector (σ) is [1.2, 0.8,

0.5, 1.0, 0.9, 1.1, 0.7, 1.3, 1.5, 0.6]. To calculate the KL divergence loss term using the formula,

1. Calculate the difference between the mean vector squared and the standard deviation vector squared: $\text{Difference} = (\mu^2) - (\sigma^2) = [-0.04, -0.25, 1.44, -0.81, -0.26, -1.21, -1.21, -1.41, -1.36, -0.09]$
2. Take the logarithm of the standard deviation vector squared: $\log(\sigma^2) = [0.78, -0.23, -0.69, 0.0, -0.11, 0.11, -0.36, 0.89, 1.18, -0.51]$
3. Add 1 to the elements obtained in step 2: $1 + \log(\sigma^2) = [1.78, 0.77, 0.31, 1.0, 0.89, 1.11, 0.64, 1.89, 2.18, 0.49]$
4. Sum the vector obtained in step 3: $\text{sum}(1 + \log(\sigma^2)) = 1.78 + 0.77 + 0.31 + 1.0 + 0.89 + 1.11 + 0.64 + 1.89 + 2.18 + 0.49 = 10.06$

Finally, calculate the KL divergence loss term:

$$\text{KL}(P \parallel Q) = -0.5 * \text{sum}(1 + \log(\sigma^2) - (\mu^2) - (\sigma^2))$$

$$= -0.5 * 10.06 = -5.03$$

So, the KL divergence loss term for this example would be -5.03.

The resulting KL divergence loss term is added to the reconstruction loss term (usually mean squared error) to obtain the total loss function that is minimized during training. The KL divergence loss term encourages the encoder to generate an approximate distribution that is close to the true prior distribution, which in turn encourages the decoder to generate more diverse and realistic data points from the latent space.

Closing note — Next, we will explore **Variational Autoencoders** and see how they leverage KL-divergence to generate new data. Your growing knowledge will make this next step even more exciting!

Data Science

Machine Learning

Artificial Intelligence

Deep Learning

AI



Follow

Written by Omkar Hankare

274 followers · 0 following

Exploring AI, ML, and data science by breaking down complex concepts into clear insights, focusing on ethical innovation and real-world applications.

No responses yet



Write a response

What are your thoughts?

More from Omkar Hankare

Very Impure



Less Impure



Pure



Decision Trees

Part 2: Information Gain

Jan 29, 2023  919  5



ally, The Gini Index is represented

$$Gini\ impurity = 1 - \sum(p$$

monly used formula is:

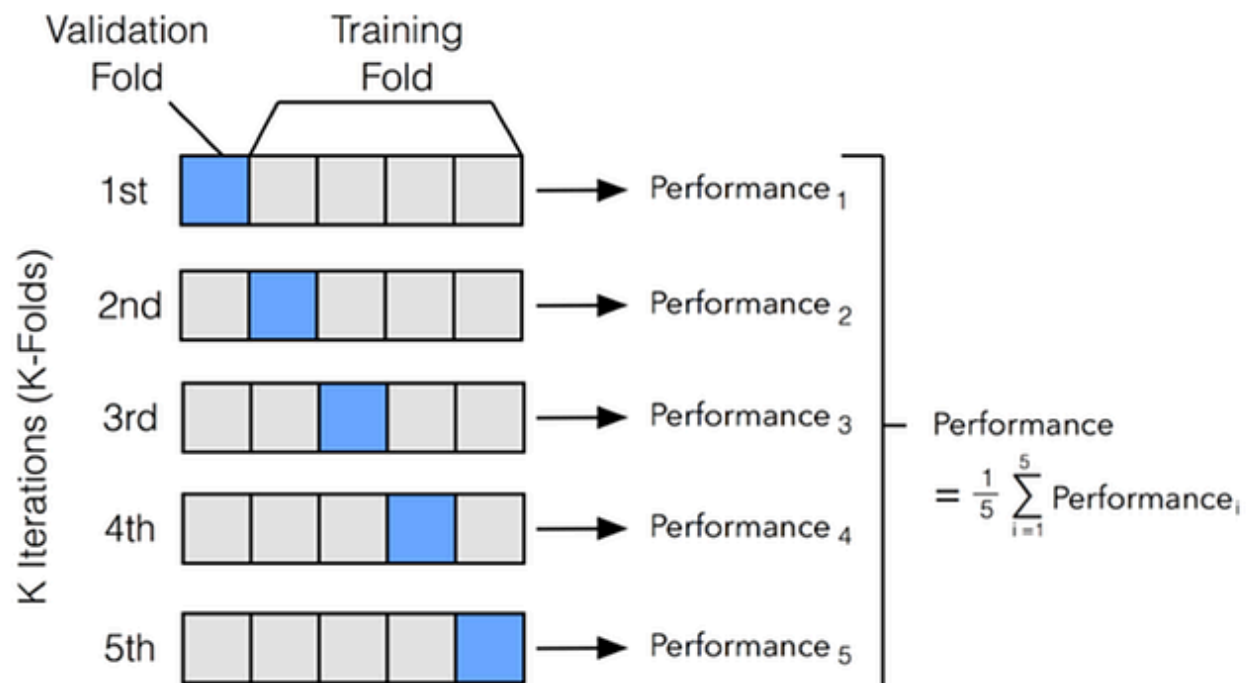
$$Gini\ impurity = 1 - \sum \left(p(i) * ($$


Decision Trees

Part 4: Gini Index

Jan 29, 2023  812  2






 Omkar Hankare

Cross Validation

Cross-validation is a technique for evaluating a machine learning model and testing its performance. Cross-validation is a technique used...

Jan 21, 2023  784



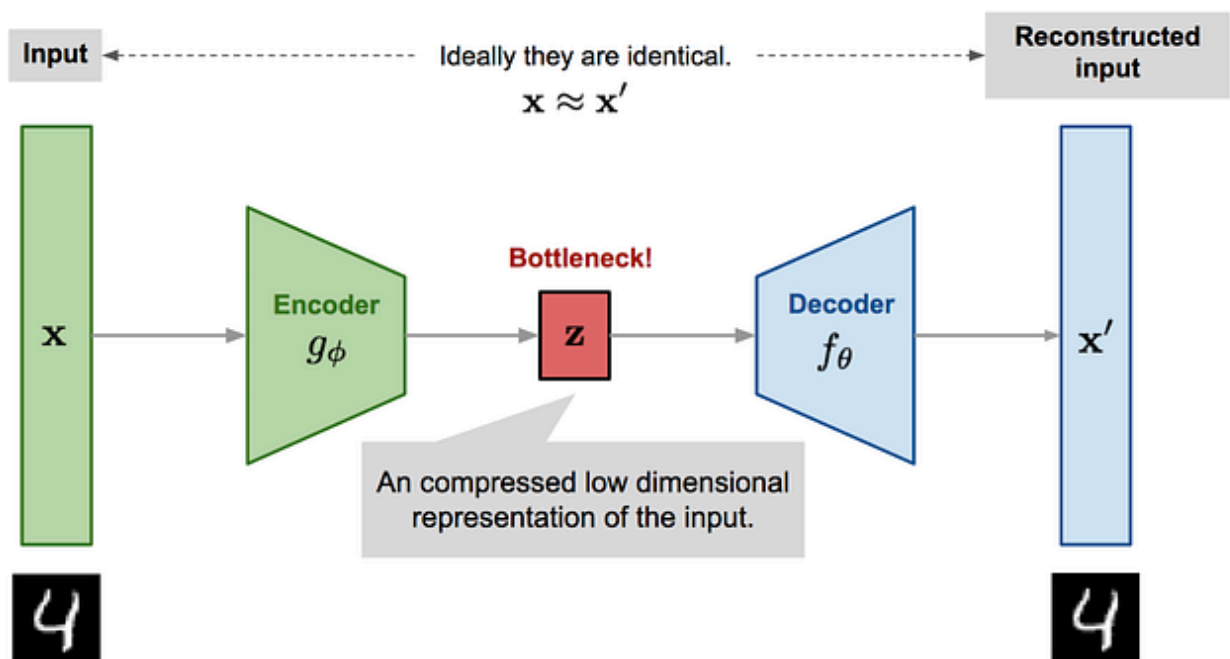
 Omkar Hankare

Convergence in deep learning

In deep learning, convergence refers to the point at which the training process reaches a stable state and the parameters of the network...

[See all from Omkar Hankare](#)

Recommended from Medium

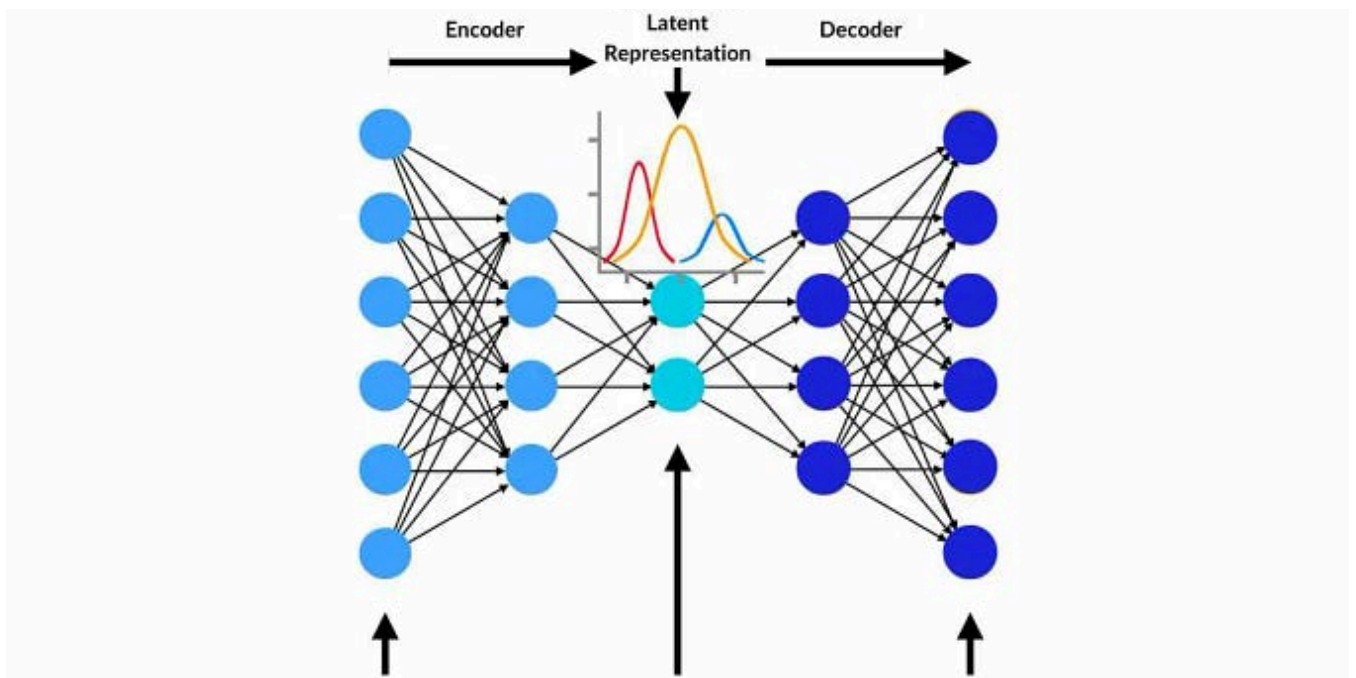


Dilip Kumar

Variational Autoencoders (VAEs) modles

1. The Foundation: The Standard Autoencoder (AE)

2d ago



DB In Data And Beyond by Anirban Bose

Math behind Variational Autoencoders (VAEs)

Sources: <https://spotintelligence.com/2023/12/27/variational-autoencoders-vae/>

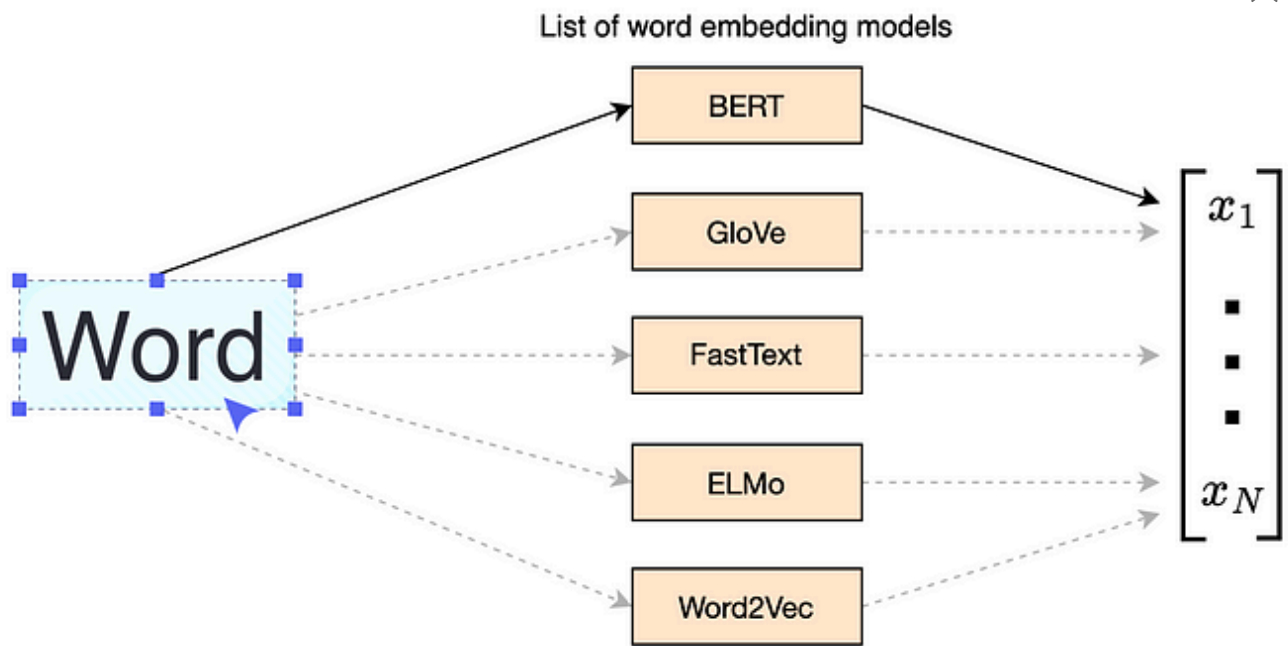
★ Apr 26 🖱 50




🔗 In Foundation Models Deep Dive by M

Parameter-Efficient Fine-Tuning for LLMs: LoRA, QLoRA, and Beyond

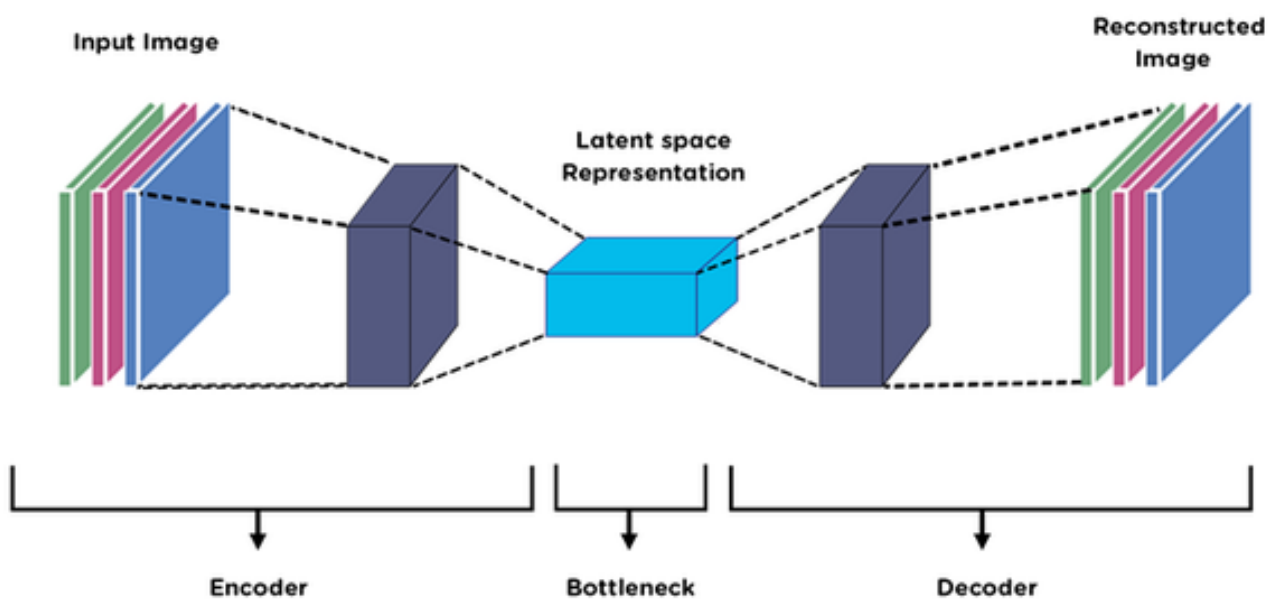
Why retrain a giant LLM when a tiny tweak will do?



 Prem Vishnoi(cloudvala)

Embedding Models Decoded: The Ultimate Guide to Transforming Words, Images & Audio into AI-Ready...

Embedding models(EM) are a handy tricks in machine learning(ML) and AI , basically tehy take raw data and turn it into numbers (vectors)...



 Himel Ghosh

Autoencoders and Transfer Learning

In the landscape of Deep Learning architectures, Autoencoders are those which provide the scope for Transfer Learning. But before jumping...

Feb 11



Original Image



VAE Re-construct



Efrat taig

VAE. The Latent Bottleneck: Why Image Generation Processes Lose Fine Details

Why do AI-generated images lose critical details? If you're frustrated by vanishing textures, blurry text, or delicate features that simply...

Mar 31



9



See more recommendations